UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/239,194	01/28/1999	JOHN S YATES JR.	5231.5-4013	9716

7590 04/29/2003

DAVID E BOUNDY
SCHULTE ROTH & ZABEL LLP
919 THIRD AVENUE
NEW YORK, NY 10022

EXAMINER
TANG, KENNETH

ART UNIT	PAPER NUMBER
2127	16

DATE MAILED: 04/29/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/239,194	YATES ET AL.	
	Examiner	Art Unit	
	Kenneth Tang	2127	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 February 2003* .

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-83* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-83* is/are rejected.

7)☒ Claim(s) *1* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *28 November 1999* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☒ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) *4, 7* .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. This office action was filed on 1/28/99.

2. A restriction requirement was made and the Applicant has traversed. After further consideration, the restriction requirement has been removed.

3. Claims 1-83 are presented for examination.

Specification

4. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

5. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

6. Claim 1 is objected to because of the following informalities:

- Incorrect grammar on line 19: "a one"

Appropriate correction is required.

Claim Rejections - 35 USC § 112

The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

7. Claim 1 is rejected under 35 U.S.C. 112, first paragraph, as containing subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. The term "exit exception" is not defined or used in the specification.

The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claim 1 recites the limitation "exit exception" on line 18. There is insufficient antecedent basis for this limitation in the claim.

9. Claim 75 recites the limitation "operating system execution mode" on lines 3-4. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 79-81 are rejected under 35 U.S.C. 102(b) as being unpatentable by Herdeg et al. (hereinafter Herdeg) (US 5,652,869).

10. Referring to claim 79, Herdig teaches a method, comprising:

- during invocation of a service routine of a computer, passing a linkage return address to the service routine at which to resume execution on completion of the service *["linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402]*;
- the linkage return address being deliberately chosen so that an attempt to execute an instruction from the linkage return address on return from the service routine will raise a program execution exception *["binary Y code", "routine", "Y exceptions", "exception conditions", col. 9, lines 1-20, and "linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", col. 6, lines 37-42]*;
- on return from the service routine, attempting to execute the instruction at the linkage return address and raising the chosen exception *["binary Y code", "routine", "Y exceptions", "exception conditions", col. 9, lines 1-20, and "linker program",*

"linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", col. 6, lines 37-42];

- after servicing the exception, returning control to a caller of the service routine *["binary Y code", "routine", "Y exceptions", "exception conditions", col. 9, lines 1-20, and "linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", "the Y routine has completed and is making a return to its X caller", col. 6, lines 37-67]*;

11. Referring to claim 80, it is rejected for the same reasons as stated in the rejection of claim 79. It is inherent that there are pointers that control the memory allocation for the method that is described in claim 79.

12. Referring to claim 81, Herdeg teaches wherein:

- the service routine is an interrupt service routine of an operating system for a computer architecture other than the architecture native to the computer *["routine", "interrupt", col. 12, lines 1-22, "architectures", "cross linker program 160", "calling and called Y object files (routines)", lines 7-41]*;
- the service routine is invoked by an asynchronous interrupt *["routine", "interrupt", col. 12, lines 1-22]*;
- the caller is coded in the instruction set native to the architecture *["routine", "interrupt", col. 12, lines 1-22, "architectures", "cross linker program 160", "calling and called Y object files (routines)", lines 7-41]*;

Claims 1, 5-7, 17, 26, 33-34, 56-57, 60, 75, and 82 are rejected under 35 U.S.C. 102(e) as being unpatentable by Nilsen et al. (hereinafter Nilsen) (US 6,081,665).

13. Referring to claims 1, 5-7, 17, 26, 33-34, 56-57, 60, 75, and 82, Nilsen teaches a method, comprising:

- without modifying a pre-existing operating system of the computer, establishing an entry exception to be raised on each entry to the operating system at a specified entry point or on a specified condition, the entry exception having an associated entry handler, the entry handler programmed to save a context of an interrupted thread and modify the thread context before delivering the modified context to the operating system *["Exception handling", "entry into such contexts", "corresponding exception", "thread state variable", col. 25, lines 42-67. It is inherent that the pre-existing operating system is not altered or modified]*;
- without modifying the operating system, establishing a resumption exception to be raised on each resumption from the operating system complementary to one of the specified entries, the resumption exception having an associated exit handler, the exit handler programmed to restore the context saved by a corresponding execution of the entry handler *["MONITOREXIT instruction (See FIG 91)", "removes the object reference", "state is saved and restored surrounding the call to the exitMonitor() function", col. 33, lines 40-67. Nilsen inherently teaches establishing a "resumption" exception by using*

variables "_psp", "_pfp", "_npsp", and "_npfp" that are each separately maintained by threads (col. 37, lines 60-67), "necessary to allow the exception thrower to search and find the appropriate exception handler whenever exceptions must be thrown", col. 25, lines 40-67. It is inherent that the pre-existing operating system is not altered or modified];

- scheduling concurrent threads of control by the operating system, each thread having an associated context, the association between a thread and a set of computer resources of the associated context being maintained by the operating system *["scheduled for execution", "dispatched thread's stack pointers (See FIG. 53 and FIG. 44)", col. 24, lines 54-58. Nilsen inherently teaches establishing a "resumption" exception by using variables "_psp", "_pfp", "_npsp", and "_npfp" that are each separately maintained by threads (col. 37, lines 60-67), "necessary to allow the exception thrower to search and find the appropriate exception handler whenever exceptions must be thrown", col. 25, lines 40-67]*;
- on detecting a specified entry to the operating system from an interrupted thread of the computer, raising and servicing the entry exception *["when an exception is raised", "invoked", "exception handler", "thread", col. 26, lines 6-15]*;
- on detecting a complementary resumption, raising and servicing the resumption exception, and returning control to the interrupted thread [*Nilsen inherently teaches establishing a "resumption" exception by using variables "_psp", "_pfp", "_npsp", and "_npfp" that are each separately maintained by threads (col. 37, lines 60-67), "necessary to allow the exception thrower to search and find the appropriate exception*

handler whenever exceptions must be thrown", col. 25, lines 40-67, "when an exception is raised", "invoked", "exception handler", "thread", col. 26, lines 6-15];

- the entry exception, exit exception, entry handler, and exit handler being cooperatively designed to maintain an association between a one of the threads and an extended context of the thread through a context change induced by the operating system, the extended context including resources of the computer associated with the thread beyond those resources whose association with the thread is maintained by the operating system *["Exception handling", "entry into such contexts", "corresponding exception", "thread state variable", col. 25, lines 42-67, "MONITOREXIT instruction (See FIG 91)", "removes the object reference", "state is saved and restored surrounding the call to the exitMonitor() function", col. 33, lines 40-67. Nilsen inherently teaches establishing a "resumption" exception by using variables "_psp", "_pfp", "_npsp", and "_npfp" that are each separately maintained by threads (col. 37, lines 60-67), "necessary to allow the exception thrower to search and find the appropriate exception handler whenever exceptions must be thrown", col. 25, lines 40-67.]*.

14. Referring to claim 75, Nilsen teaches the step of setting of a register to a value that specifies actions to be taken by an exception handler invoked on the transition to convert operands from one form to another to conform to a data storage convention of the operating system mode *["registers", "values", "pointers", col. 36, lines 41-49, and "translations of exception handling context", "contents of those registers are saved and restored", "entry into the exception handling context", col. 77, lines 36-43]*. In addition, it has already been

established that Nilsen in view of Kukol teaches the conversion from one form to another (different instruction sets). Because of this, it is inherent that settings of a register to values that specify conversion actions are used to perform conversion actions.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claim 2, 8-10, 20-22, 35-36, 40-42, 58, and 63-65 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Richter et al. (hereinafter Richter) (US 5,481,684).

15. Referring to claim 2, Nilsen fails to explicitly teach the method wherein the operating system is an operating system for a computer architecture other than the architecture native to the computer. However, Richter teaches this limitation with the example that "RISC instruction code may reside within a CISC segment. The CISC architecture is extended to provide segments that can hold RISC code or CISC code" *[col. 1, lines 60-66 through col. 2, lines 1-43, and see Abstract]*. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of having a computer architecture other than the native architecture to the existing method for the reason of increasing compatibility. For example, Richter teaches that an enormous base of software has been written for existing operating

systems such as DOS and MS WINDOWS. However, these operating systems can only be operated on x86 microprocessors. The x86 architecture is an old complex instruction set computer (CISC) architecture and is quite different from today's highly optimized reduced instruction set computers (RISCs). But with the emulation programs, microprocessors can translatex86 CISC instructions to RISC instructions so that one architecture could utilize any software from the other.

16. Referring to claims 8, 20, 35, 40, 58, and 63, it is rejected for the same reasons as stated in the rejection of claim 2. Further, it is inherent that there are tasks and threads that are scheduled in operating systems.

17. Referring to claims 9, 21, 41, and 64, Richter teaches wherein the computer additionally executes an operating system native to the computer, and each exception is classified for handling by one of the two operating systems *["dual instruction set", "exception", "CISC", "RISC", "operating systems", col. 3, lines 1-67].*

18. Referring to claims 10, 22, 36, 42, and 65, Richter teaches the method of claim 8, wherein operating system and the interrupted thread execute in different instruction set architectures of the computer *["A signal is needed to cause the processor to switch the instruction set being decoded. An exception or interrupt can provide this signal, or a separate instruction can be defined to switch instruction sets. Jumping from a CISC user program to another segment written in RISC code without signaling an interrupt or exception cold cause*

unpredictable results or even a system crash unless a method is employed to trigger the switch to RISC decoding.", col. 5, lines 55-67].

Claims 12-13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Yates et al. (hereinafter Yates) (US 5,802,373) and further in view of Richter et al. (hereinafter Richter) (US 5,481,684).

19. Referring to claim 12, Nilsen teaches establishing exceptions (as mentioned earlier). Yates teaches a dual instruction sets *["non-native", "native", "instructions", see Abstract].* Nilsen in view of Yates fails to explicitly and specifically teach each exception is classified for handling by one of the two operating systems. However, Richter teaches wherein operating system and the interrupted thread execute in different instruction set architectures of the computer *["A signal is needed to cause the processor to switch the instruction set being decoded. An exception or interrupt can provide this signal, or a separate instruction can be defined to switch instruction sets. Jumping from a CISC user program to another segment written in RISC code without signaling an interrupt or exception cold cause unpredictable results or even a system crash unless a method is employed to trigger the switch to RISC decoding.", col. 5, lines 55-67].* It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of classifying the exception for handling by one of the two operating systems to the existing method for the reason of increasing the control so that two instruction sets or architectures can be utilized.

20. Referring to claim 13, it is rejected for the same reasons as stated in the rejection of claim 2.

Claim 3, 45-48, 61, 68-70 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Blomgren (US 5,598,546).

21. Referring to claims 3 and 68, Nilsen fails to explicitly teach wherein the operating-system-maintained resources of the thread context include data registers of the non-native computer architecture, the method further comprising:

- modifying at least half of the data registers of the portion of the thread context maintained by the operating system before delivering the thread to the non-native operating system.

However, Blomgren teaches a system that can access and modify at least half of the data registers, specifically, 8 or 16 bits of a 32-bit register *["CISC mode", "low half", col. 17, lines 43-55]*. It is inherent those modifications are done before delivery to the non-native operating system because all of this is done before the function calls for a return address. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of modifying at least half of the data registers before delivering the thread to the non-native operating system for the reason of improving data flow control. Data in a registry is categorized and separated to give restricted access towards certain information.

22. Referring to claim 45, it is rejected for the same reasons as stated in the rejection of claim 3. Further, it is inherent that a service routine is used to perform this function.

23. Referring to claims 46 and 69, it is rejected for the same reasons as stated in the rejection of claim 29.

24. Referring to claims 47-48 and 70, Blomgren teaches a system that can access/modify/overwrite at least half of the data registers, specifically, 8 or 16 bits of a 32-bit register *["CISC mode", "low half", col. 17, lines 43-55]*. Nilsen teaches checking the validity *["checks first to see if this exception handler desires to handle the thrown exception. If so, we handle it here. If not, we simply throw the exception to the surrounding exception handler", col. 26, lines 5-15]*.

25. Referring to claim 61, it is rejected for the same reasons as stated in the rejection of claim 3.

Claims 4, 15-16, 23-25, 38, 43-44, 59, 66-67 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Kukol (US 5,628,016).

26. Referring to claims 4, 23, 43, and 66, Nilsen teaches the method wherein thread scheduler and the thread execute in different instruction sets of the computer but the reference fails to explicitly teach the entry and exit exception are automatically invoked, without explicit software request, on a transition between the thread instruction set and the operating system instruction set. However, Kukol teaches this automatically invoked exception handling *["handling of exceptions", "automatic", "exception-handling problem", automatically", "constructors", "destructors", col. 6, lines 50-67 and col. 13, lines 40-55, and col. 14, lines 28-67]*. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of automatically invoking the exception handling to the existing method for the reason of improving system convenience through automation. Dynamic allocation prevents the program from manually having to do this, which requires a lot more work.

27. Referring to claim 15, it is rejected for the same reasons as stated in the rejection of claim 4.

28. Referring to claims 16, 24, 44, and 67, Nilsen teaches the method of claim 15, wherein the thread execution mode and the thread scheduler execution mode are two different instruction set architectures of the computer *[e.g. "dispatcher" operates between modes, "watchdog", "schedule", "thread", "execution", "task", "PERC virtual machine", col. 37, lines 15-67, and see Abstract]*.

29. Referring to claim 25, Nilsen teaches the step of setting of a register to a value that specifies actions to be taken by an exception handler invoked on the transition to convert operands from one form to another to conform to a data storage convention of the thread scheduler execution mode *["registers", "values", "pointers", col. 36, lines 41-49, and "translations of exception handling context", "contents of those registers are saved and restored", "entry into the exception handling context", col. 77, lines 36-43].* In addition, it has already been established that Nilsen in view of Kukol teaches the conversion from one form to another (different instruction sets). Because of this, it is inherent that settings of a register to values that specify conversion actions are used to perform conversion actions.

30. Referring to claim 38, it is rejected for the same reasons as stated in the rejection of claim 4.

31. Referring to claim 59, it is rejected for the same reasons as stated in the rejection of claim 4.

Claims 11, 18, 30, 39, 50, 62, 71, and 74 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Yates et al. (US 5,802,373).

32. Referring to claim 11, Nilsen fails to explicitly teach the method wherein the operating system is in a binary code for a computer architecture non-native to the architecture of the computer. However, Yates teaches a computer architecture non-native instructions being fed to a binary translator *["non-native instruction", "binary translator", "non native computer system", "native instruction routine", see Abstract]*. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of having binary code for a computer architecture non-native to the architecture to the computer for the reason of improving communication. All computers read binary code at the lower level.

33. Referring to claims 18, 30, 39, 50, 62, 71, Nilsen teaches exception handlers for maintaining an association between one of the threads and an extended context of the thread through a context change induced by the operating system. Nilsen fails to explicitly teach modifying the return address for the linkage/association. However, Yates teaches that return addresses can be altered, changed or modified *["changed", "return address", "modify the non-native return address", col. 27, lines 64-67 through col. 28, lines 1-4, and col. 29, lines 50-65]*. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of changing the return address to the existing method for the reason of increasing the flexibility of the program. If return addresses can be altered, different addresses other than the original one can be returned, which can provide more functionality in the program.

34. Referring to claim 74, Nilsen teaches the method wherein the linkage register is modified with information indicating a storage location at which at least the portion of the thread context

to be modified is saved before the modifying *["Exception handling", "entry into such contexts", "corresponding exception", "thread state variable", col. 25, lines 42-67.* Nilsen teaches saving information before they are modifying, *"adjusts the stack and other state information as necessary", "routine saves the offset of the old frame and stack pointers in local C variables", "saves the current value of the C stack pointer", "machine registers", col. 15, lines 12-58].*

Claims 14, 19, 31, 51, and 73 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Yates et al. (US 5,802,373) and further in view of Blomgren (US 5,598,546).

Referring to claim 14, Nilsen in view of Yates fails to explicitly teach wherein the resources of the context maintained in association with the thread by the non-native operating system include data registers of the non-native computer architecture, the method further comprising:

- in the entry exception handler, modifying at least half of the data registers of the portion of the process context maintained by the non-native operating system before delivering the process to the non-native operating system, at least some of the modified registers being redundantly written with data to enable checking of the validity of the contents of the context in the resumption exception handler.

Nilsen teaches checking the validity *["checks first to see if this exception handler desires to handle the thrown exception. If so, we handle it here. If not, we simply throw the exception to*

the surrounding exception handler", col. 26, lines 5-15]. Nilsen fails to explicitly teach in the entry exception handler, modifying at least half of the data registers of the portion of the process context maintained by the non-native operating system before delivering the process to the non-native operating system. However, Blomgren teaches a system that can access and modify at least half of the data registers, specifically, 8 or 16 bits of a 32-bit register *["CISC mode", "low half", col. 17, lines 43-55].* Blomgren also teaches having a dual architecture pipeline It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of modifying at least half of the data registers before delivering the thread to the non-native operating system for the reason of improving data flow control. Data in a registry is categorized and separated to give restricted access towards certain information.

35. Referring to claim 19, it is rejected for the same reasons as stated in the rejection of claim 3.

36. Referring to claim 31, it is rejected for the same reasons as stated in the rejection of claim 3.

37. Referring to claim 51, it is rejected for the same reasons as stated in the rejection of claim 3.

38. Referring to claim 73, it is rejected for the same reasons as stated in the rejection of claim 3.

Claims 27 and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Blomgren (US 5,598,546), and further in view of Richter et al. (hereinafter Richter) (US 5,481,684).

39. Referring to claim 27, it is rejected for the same reasons as stated in the rejection of claim 3.

40. Referring to claim 29, Blomgren teaches modifying registers (previously shown). Nilsen teaches saving information before they are modifying *["adjusts the stack and other state information as necessary", "routine saves the offset of the old frame and stack pointers in local C variables", "saves the current value of the C stack pointer", "machine registers", col. 15, lines 12-58]*. It is notoriously well known in the art that storage location of data/information can be overwritten/saved in registers. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of saving the storage location before modifying the registers for the reason of increasing the control of the program by preventing the loss of any important original/previous data values.

Claims 28 is rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Blomgren (US 5,598,546), and further in view

of Richter et al. (hereinafter Richter) (US 5,481,684), and futher in view of Yates et al. (hereinafter Yates) (US 5,802,373).

41. Referring to claim 28, Blomgren teaches modifying registers (previously shown). Nilsen in view of Blomgren, and further in view of Richter fails to explicitly teach wherein at least some of the modified registers are overwritten by a timestamp *["time stamp", col. 7, lines 50-67]*. However, Yates teaches that a timestamp can be information that can be stored and associated with a file. It is also inherent that when this information is stored, it is done in data registers. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of storing a timestamp to modify a register for the reason of increasing the function of the program by being able to track time with a receipt or audit trail.

Claim 32, 54-55, 76-78 is rejected under 35 U.S.C. 103(a) as being obvious over Nilsen et al. (hereinafter Nilsen) (US 6,081,665).

42. Referring to claims 32, 54-55, and 76-77, Nilsen teaches either the step of deferring delivery of an interrupt before interrupting the thread by a time sufficient *["interrupt trigger", "dispatcher", "priority", "executing task", "watchdog", "PERC", "asynchronous interrupts from the watchdog task and the alarm timer", "the dispatcher can only use its PERC stacks during times when it is sure that the most recently dispatched PERC task is blocked and/or*

suspended", "scheduled for execution block", "awakened", col. 37, lines 15-67 through col. 38, lines 1-17 and col. 76, lines 1-28].

Nilsen fails to explicitly teach checkpoints. However, it is well known in the art that thresholds or checkpoints can be used as a point of transition. For example, a threshold could be placed to indicate where context is reduced. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of a checkpoint where the interrupt reaches to the existing method for the reason of improving control of the program by having a marker or indicating point where there is a change of state.

43. Referring to claim 78, Nilsen teaches the method further comprising either the step of storing at least the extended context, being the resources beyond those whose resource association with the thread is maintained by the operating system, into a storage location, a pool of storage locations being managed by a queuing discipline in which empty storage locations in which a context is to be saved are allocated from the head of the queue, recently-emptied storage locations for reuse are enqueued at the head of the queue, and full storage locations to be saved are queued at the tail of the queue [*list queue", "waiting_list queue", col. 35, lines 28-67].* It is also notoriously well known in the art that a queue data structure could be used to contain this pool of storage locations. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include a data structure such as a queue to contain the storage of the extended context for the reason of improving data control data. Queues are one of many basic data structures that allows different operations to be performed on the data.

44.

Claim 37 is rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Richter et al. (hereinafter Richter) (US 5,481,684), and further in view of Blomgren (US 5,598,546).

45. Referring to claim 37, Nilsen in view of Richter fails to explicitly teach wherein the resources of the context maintained in association with the thread by the non-native operating system include data registers of the non-native computer architecture, the method further comprising:

- in the entry exception handler, modifying at least half of the data registers of the portion of the process context maintained by the non-native operating system before delivering the process to the non-native operating system, at least some of the modified registers being redundantly written with data to enable checking of the validity of the contents of the context in the resumption exception handler.

Nilsen teaches checking the validity *["checks first to see if this exception handler desires to handle the thrown exception. If so, we handle it here. If not, we simply throw the exception to the surrounding exception handler", col. 26, lines 5-15].* Nilsen fails to explicitly teach in the entry exception handler, modifying at least half of the data registers of the portion of the process context maintained by the non-native operating system before delivering the process to the non-native operating system. However, Blomgren teaches a system that can access and modify at least half of the data registers, specifically, 8 or 16 bits of a 32-bit register *["CISC mode",*

"low half", col. 17, lines 43-55]. Blomgren also teaches having a dual architecture pipeline It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of modifying at least half of the data registers before delivering the thread to the non-native operating system for the reason of improving data flow control. Data in a registry is categorized and separated to give restricted access towards certain information.

Claim 49 is rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Blomgren (US 5,598,546), and further in view of Yates et al. (hereinafter Yates) (US 5,802,373).

46. Referring to claim 49, Blomgren teaches modifying registers (previously shown). Nilsen in view of Blomgren fails to explicitly teach wherein at least some of the modified registers are overwritten by a timestamp. However, Yates teaches that a timestamp can be information that can be stored and associated with a file *["time stamp", col. 7, lines 50-67].* It is also inherent that when this information is stored, it is done in data registers. It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of storing a timestamp to modify a register for the reason of increasing the function of the program by being able to track time with a receipt or audit trail.

Claims 52-53 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Yates et al. (hereinafter Yates) (US 5,802,373), and further in view of Herdeg et al. (hereinafter Herdeg) (US 5,652,869).

47. Referring to claim 52, Nilsen fails to explicitly teach the method further comprising:

- as part of servicing the entry exception, modifying a linkage return address of the interrupted process, the return address being deliberately chosen so that an attempt to execute an instruction from the return address on return from the operating system will raise the resumption exception

However, Herdig teaches a method, comprising:

- during invocation of a service routine of a computer, passing a linkage return address to the service routine at which to resume execution on completion of the service *["linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402]*;
- the linkage return address being deliberately chosen so that an attempt to execute an instruction from the linkage return address on return from the service routine will raise a program execution exception *["binary Y code", "routine", "Y exceptions", "exception conditions", col. 9, lines 1-20, and "linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", col. 6, lines 37-42]*;
- on return from the service routine, attempting to execute the instruction at the linkage return address and raising the chosen exception *["binary Y code", "routine", "Y*

exceptions", "exception conditions", col. 9, lines 1-20, and "linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", col. 6, lines 37-42];

- after servicing the exception, returning control to a caller of the service routine *["binary Y code", "routine", "Y exceptions", "exception conditions", col. 9, lines 1-20, and "linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", "the Y routine has completed and is making a return to its X caller", col. 6, lines 37-67]*;

It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of modifying a linkage return address of the interrupted process, the return address being deliberately chosen so that an attempt to execute an instruction from the return address on return from the operating system will raise the resumption exception to the existing method for the reason of improving data flow control and increasing the flexibility of the program. If return addresses can be altered, different addresses other than the original one can be returned, which can provide more functionality in the program.

48. Referring to claim 53, it is rejected for the same reasons as stated in the rejection of 52. It is inherent that there are pointers that control the memory allocation for the method that is described in claim 52.

Claim 72 is rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Yates et al. (hereinafter Yates) (US 5,802,373), and further in view of Herdeg et al. (hereinafter Herdeg) (US 5,652,869).

49. Referring to claim 72, Nilsen in view of Yates fails to explicitly teach the method wherein the linkage register is modified with information indicating an execution path by which, or a condition on which, execution arrived at the entry handler. However, Herdeg teaches the use of conditions for the execution of X and Y programs *["conditions", "X", "Y", col. 6, lines 37-41, and col. 9, lines 1-22, and col. 13, lines 25-40].* It would have been obvious to one of ordinary skill in the art at the time the invention was made to include the feature of having the linkage register being modified with information indicating an execution path by which, or a condition on which, execution arrived at the entry handler to the existing method for the reason of increasing the control of the program by allowing data state processing from condition statements.

Claim 83 is rejected under 35 U.S.C. 103(a) as being unpatentable over Nilsen et al. (hereinafter Nilsen) (US 6,081,665) in view of Richter et al. (hereinafter Richter) (US 5,481,684) and further in view of Herdeg et al. (hereinafter Herdeg) (US 5,652,869).

50. Referring to claim 83, Nilsen fails to explicitly and specifically teach further comprising the step of:

- during invocation of a service routine of the operating system, passing a linkage return address to the service routine at which to resume execution on completion of the service, the linkage return address being deliberately chosen so that an attempt to execute an instruction from the linkage return address on return from the service routine will raise a program execution exception;
- on return from the service routine, attempting to execute the instruction at the linkage return address and raising the chosen exception;
- after servicing the exception, returning control to a caller of the service routine.

However, Herdig teaches a method, comprising:

- during invocation of a service routine of a computer, passing a linkage return address to the service routine at which to resume execution on completion of the service *["linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402]*;
- the linkage return address being deliberately chosen so that an attempt to execute an instruction from the linkage return address on return from the service routine will raise a program execution exception *["binary Y code", "routine", "Y exceptions", "exception conditions", col. 9, lines 1-20, and "linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", col. 6, lines 37-42]*;
- on return from the service routine, attempting to execute the instruction at the linkage return address and raising the chosen exception *["binary Y code", "routine", "Y exceptions", "exception conditions", col. 9, lines 1-20, and "linker program",*

"linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", col. 6, lines 37-42];

- after servicing the exception, returning control to a caller of the service routine *["binary Y code", "routine", "Y exceptions", "exception conditions", col. 9, lines 1-20, and "linker program", "linkages", "routines", col. 8, lines 19-24, and "execution routines", see col. 13, lines 25-40 and Fig. 8, item 402, "return address", "the Y routine has completed and is making a return to its X caller", col. 6, lines 37-67]*;

It would have been obvious to one of ordinary skill in the art at the time the invention was made to include these mentioned features of the linkage return address to the existing method for the reason of improving communication between the associations. More linking features will allow for stronger control of the associations.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kenneth Tang whose telephone number is (703) 305-5334. The examiner can normally be reached on 9:00am-6:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Follansbee can be reached on (703) 305-8498. The fax phone numbers for the organization where this application or proceeding is assigned are (703) 746-7239 for regular communications and (703) 746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is none.

kt
April 21, 2003

JOHN FOLLANSBEE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100